Tree cheatsheet for coding interviews

👋

Introduction

A tree is a widely used abstract data type that represents a hierarchical structure with a set of connected nodes. Each node in the tree can be connected to many children, but must be connected to exactly one parent, except for the root node, which has no parent.
A tree is an undirected and connected acyclic graph. There are no cycles or loops. Each node can be like the root node of its own subtree, making recursion a useful technique for tree traversal.
For the purpose of interviews, you will usually be asked on binary trees as opposed to ternary (3 children) or N-ary (N children) trees. In this page,we will cover binary trees and binary search trees, which is a special case of binary trees.
Trees are commonly used to represent hierarchical data, e.g. file systems, JSON, and HTML documents. Do check out the section on Trie, which is an advanced tree used for efficiently storing and searching strings.

Learning resources

- Videos
    - Trees, University of California San Diego
    - A Brief Guide to Binary Search Trees (slides), Samuel Albanie, University of Cambridge
    - A Brief Guide to Red-Black Trees (slides), Samuel Albanie, University of Cambridge
    - A Brief Guide to B-trees (slides), Samuel Albanie, University of Cambridge
- Readings
    - How To Not Be Stumped By Trees, basecs

    - Leaf It Up To Binary Trees, basecs
- Additional (only if you have time)
    - The Little AVL Tree That Could, basecs
    - Busying Oneself With B-Trees, basecs
    - Painting Nodes Black With Red-Black Trees, basecs

Common terms you need to know

- Neighbor - Parent or child of a node
- Ancestor - A node reachable by traversing its parent chain
- Descendant - A node in the node's subtree
- Degree - Number of children of a node
- Degree of a tree - Maximum degree of nodes in the tree
- Distance - Number of edges along the shortest path between two nodes
- Level/Depth - Number of edges along the unique path between a node and the root node
- Width - Number of nodes in a level

Binary tree

Binary means two, so nodes in a binary trees have a maximum of two children.
Binary tree terms

- Complete binary tree - A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible.
- Balanced binary tree - A binary tree structure in which the left and right subtrees of every node differ in height by no more than 1.

Traversals
Given such a tree, these are the results for the various traversals.

- In-order traversal - Left -> Root -> Right
    - Result: 2, 7, 5, 6, 11, 1, 9, 5, 9
- Pre-order traversal - Root -> Left -> Right
    - Result: 1, 7, 2, 6, 5, 11, 9, 9, 5
- Post-order traversal - Left -> Right -> Root

    - Result: 2, 5, 11, 6, 7, 5, 9, 9, 1

Note that in-order traversal of a binary tree is insufficient to uniquely serialize a tree. Pre-order or post-order traversal is also required.

Binary search tree (BST)

In-order traversal of a BST will give you all elements in order.
Be very familiar with the properties of a BST and validating that a binary tree is a BST. This comes up more often than expected.
When a question involves a BST, the interviewer is usually looking for a solution which runs faster than O(n).

Time complexity

Operation	Big-O
Access	O(log(n))
Search	O(log(n))
Insert	O(log(n))
Remove	O(log(n))

Space complexity of traversing balanced trees is O(h) where h is the height of the tree, while traversing very skewed trees (which is essentially a linked list) will be O(n).

Things to look out for during interviews

You should be very familiar with writing pre-order, in-order, and post-order traversal recursively. As an extension, challenge yourself by writing them iteratively. Sometimes interviewers ask candidates for the iterative approach, especially if the candidate finishes writing the recursive approach too quickly.

Corner cases

- Empty tree
- Single node

- Two nodes
- Very skewed tree (like a linked list)

Common routines

Be familiar with the following routines because many tree questions make use of one or more of these routines in the solution:

- Insert value
- Delete value
- Count number of nodes in tree
- Whether a value is in the tree
- Calculate height of the tree
- Binary search tree
    - Determine if is binary search tree
    - Get maximum value
    - Get minimum value

Techniques

Use recursion

Recursion is the most common approach for traversing trees. When you notice that the subtree problem can be used to solve the entire problem, try using recursion.
When using recursion, always remember to check for the base case, usually where the node is `null`.
Sometimes it is possible that your recursive function needs to return two values.

Traversing by level

When you are asked to traverse a tree by level, use breadth-first search.

Summation of nodes

If the question involves summation of nodes along the way, be sure to check whether nodes can be negative.

Essential questions

These are essential questions to practice if you're studying for this topic.

- Binary Tree
    - Maximum Depth of Binary Tree
    - Invert/Flip Binary Tree
- Binary Search Tree
    - Lowest Common Ancestor of a Binary Search Tree

Recommended practice questions

These are recommended questions to practice after you have studied for the topic and have practiced the essential questions.

- Binary tree
    - Same Tree
    - Binary Tree Maximum Path Sum
    - Binary Tree Level Order Traversal
    - Lowest Common Ancestor of a Binary Tree
    - Binary Tree Right Side View
    - Subtree of Another Tree
    - Construct Binary Tree from Preorder and Inorder Traversal
    - Serialize and Deserialize Binary Tree
- Binary search tree
    - Validate Binary Search Tree
    - Kth Smallest Element in a BST